Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik

INSTITUT FÜR THEORETISCHE INFORMATIK

Entfernen von ϵ-Übergängen

Karlsruher Institut für Technologie

Satz:
Zu jedem NEA $\mathcal{A}$ mit ε-Übergängen gibt es einen NEA $\widetilde{\mathcal{A}}$ ohne ε-Übergänge, der dieselbe Sprache akzeptiert und nicht mehr Zustände hat.

Entfernen von ϵ-Übergängen

Satz:
Zu jedem NEA $\mathcal{A}$ mit ε-Übergängen gibt es einen NEA $\widetilde{\mathcal{A}}$ ohne ε-Übergänge, der dieselbe Sprache akzeptiert und nicht mehr Zustände hat.

Beweis: Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ ein NEA mit ε–Übergängen.
Wir konstruieren $\widetilde{\mathcal{A}} := (\widetilde{Q}, \Sigma, \widetilde{\delta}, \widetilde{s}, \widetilde{F})$ wie folgt:

- $\widetilde{Q} := (Q \backslash F) \cup \widetilde{F}$
- $\widetilde{s} := s$
- $$\widetilde{\delta}(q, a) = \begin{cases} \{q\} & \text{falls } a = \varepsilon \\ \delta(E(q), a) & \text{sonst} \end{cases}$$
- $\widetilde{F} := \{q \mid E(q) \cap F \neq \emptyset\}$

Damit akzeptiert $\widetilde{A}$ dieselbe Sprache wie $\mathcal{A}$, und $|\widetilde{Q}| \leq |Q|$.

EA $\rightarrow$ Regularität

Satz:
Jede Sprache, die von einem endlichen Automaten erkannt wird, ist regulär.

Beweis: EA $\rightarrow$ Regularität

- Sei DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ gegeben.
- Es ist zu zeigen, dass $L(\mathcal{A})$ regulär ist.

Es gilt:

$L = \{w \in \Sigma^* \mid \mathcal{A}$ endet nach Abarbeitung von w in einem Zustand aus $F\}$

- Die Abarbeitung eines Wortes $w = a_1 \dots a_k$ bewirkt das Durchlaufen einer Folge von Zuständen $s, q_1, \dots, q_k$, wobei nicht notwendig $q_i \neq q_j$ für $i \neq j$ gilt.
- Wir suchen die Wörter, so dass der letzte Zustand in F ist.
- Betrachte für jeden Zustand $f \in F$ getrennt die Wörter, deren Abarbeitung in f endet.

Beweis: EA $\rightarrow$ Regularität

- Sei DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ gegeben.
- Es ist zu zeigen, dass $L(\mathcal{A})$ regulär ist.

Es gilt:

$L = \{w \in \Sigma^* \mid \mathcal{A}$ endet nach Abarbeitung von w in einem Zustand aus $F\}$

Zu $f \in F$ definiere:

$$\begin{aligned} L_f &:= \{w \in \Sigma^* \mid \mathcal{A} \text{ endet nach Abarbeitung von } w \text{ in } f\} \\ &= \{w \in \Sigma^* \mid w \text{ überführt } s \text{ in } f \text{ (im Automaten } \mathcal{A})\} \end{aligned}$$

- Damit ist $L = \bigcup_{f \in F} L_f$.
- Wenn wir zeigen können, dass für alle $f \in F$ die Sprache L_f regulär ist, so ist auch L regulär.

Beweis: EA $\rightarrow$ Regularität

$$L_f := \{w \in \Sigma^* \mid w \text{ überführt } s \text{ in } f \text{ (im Automaten } \mathcal{A})\}$$

Ab jetzt sei $Q = \{q_1, \ldots, q_n\}$.
Wir definieren zu

$$q_r, q_t \in Q\colon L_{q_r,q_t} := \{w \in \Sigma^* \mid w \text{ überführt } q_r \text{ in } q_t\} \ .$$

Insbesondere gilt also: $L_f = L_{s,f}$. Unterteile L_{q_r,q_t}:

$$L_{q_r,i,q_t} := \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} \text{Abarbeitung von } w \text{ aus } q_r \text{ nach } q_t \text{ hat nur} \\ \text{Zwischenzustände } \{q_1, \ldots, q_i\} \end{array} \right\}$$

(also w bewirkt: $q_r \rightarrow \underbrace{\ldots\ldots\ldots}_{\in\{q_1,\ldots,q_i\}} \rightarrow q_t$.)

Damit gilt $L_{q_r,q_t} = L_{q_r,n,q_t}$.

Beweis: EA $\rightarrow$ Regularität

$$L_{q_r,i,q_t} := \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l}\text{Abarbeitung von } w \text{ aus } q_r \text{ nach } q_t \text{ hat nur} \\ \text{Zwischenzustände } \{q_1, \ldots, q_i\}\end{array} \right\}$$

Wir zeigen, dass L_{q_r,i,q_t} für $q_r, q_t \in Q$ und $1 \leq i \leq n$ regulär sind:

- Zunächst betrachten wir direkte Überführungen, also $i = 0$:

  $$L_{q_r,0,q_t} := \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l}\text{Abarbeitung von } w \text{ führt von } q_r \text{ nach } q_t \\ \text{ohne Zwischenzustand}\end{array} \right\}$$

  Falls $r = t$ und somit $q_r = q_t$ ist, ist

  $$L_{q_r,0,q_t} = \{a \in \Sigma \mid \delta(q_t, a) = q_t\} \cup \{\varepsilon\} \ .$$

  Andernfalls betrachten wir alle w mit $q_r \stackrel{w}{\rightarrow} q_t$, ohne Zwischenzustände, also

  $$L_{q_r,0,q_t} = \{a \in \Sigma \mid \delta(q_r, a) = q_t\} \ .$$

  Diese Sprachen sind jeweils regulär.

Beweis: EA $\rightarrow$ Regularität

- Betrachte nun $i = 1$:

$$L_{q_r,1,q_t} := \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ überführt } q_r \text{ in } q_t \text{ entweder direkt oder} \\ \text{unter Benutzung nur von } q_1 \end{array} \right\}$$

  Es gilt dann:

$$L_{q_r,1,q_t} = L_{q_r,0,q_t} \cup \left(L_{q_r,0,q_1} \cdot L^*_{q_1,0,q_1} \cdot L_{q_1,0,q_t} \right)$$

  Also ist $L_{q_r,1,q_t}$ auch wieder regulär.

- Es gilt allgemein:

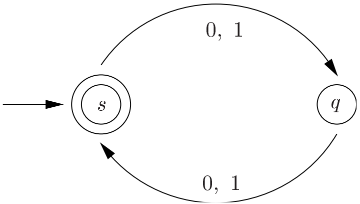
$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} \left(L_{q_{i+1},i,q_{i+1}} \right)^* L_{q_{i+1},i,q_t} \right)$$

Beweis: EA $\rightarrow$ Regularität

- Es wurden für $L_{\cdot,i+1,\cdot}$ nur die Sprachen $L_{\cdot,i,\cdot}$ und $\cup, \cdot, *$ verwendet.
- Damit ist gezeigt (per Induktion), dass $L_{\cdot,i+1,\cdot}$ regulär ist für beliebiges i $(1 \leq i+1 \leq n)$ und alle Zustandspaare aus Q^2.
- Damit ist gezeigt, dass insbesondere $L_f = L_{s,n,f}$ regulär ist für jedes $f \in F$.

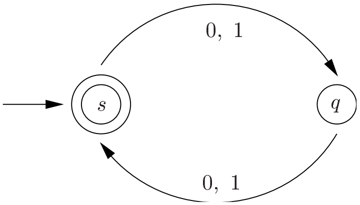
Beispiel

Sei $(Q, \Sigma, \delta, s, F)$ mit $Q := \{q_1 := s, q_2 := q\}, \Sigma := \{0, 1\}, F := \{s\}$



Gesucht: $L(Q, \Sigma, \delta, s, F)$. Es gilt $L = L_{q_1, 2, q_1}$.

Beispiel



Gesucht: $L(Q, \Sigma, \delta, s, F)$. Es gilt $L = L_{q_1,2,q_1}$.

Dann ist

$$L_{q_i,0,q_i} = \varepsilon$$

$$L_{q_i,0,q_j} = (0 \cup 1) \text{ für } i, j \in \{1, 2\}, i \neq j$$

$$\begin{aligned}
L_{q_1,1,q_1} &= L_{q_1,0,q_1} \cup L_{q_1,0,q_1} \left(L_{q_1,0,q_1}\right)^* L_{q_1,0,q_1} = \varepsilon \\
L_{q_1,1,q_2} &= L_{q_1,0,q_2} \cup L_{q_1,0,q_1} (L_{q_1,0,q_1})^* L_{q_1,0,q_2} = (0 \cup 1) \cup \varepsilon\varepsilon^* (0 \cup 1) = 0 \cup 1 \\
L_{q_2,1,q_1} &= (0 \cup 1) \cup (0 \cup 1)\varepsilon^*\varepsilon = 0 \cup 1 \\
L_{q_2,1,q_2} &= \varepsilon \cup (0 \cup 1)\varepsilon^*(0 \cup 1) = \varepsilon \cup (0 \cup 1)(0 \cup 1)
\end{aligned}$$

$$\begin{aligned}
L = L_{q_1,2,q_1} &= L_{q_1,1,q_1} \cup \left(L_{q_1,1,q_2}(L_{q_2,1,q_2})^* L_{q_2,1,q_1}\right) \\
&= \varepsilon \cup (0 \cup 1) \left((0 \cup 1)(0 \cup 1)\right)^* (0 \cup 1) = \left((0 \cup 1)(0 \cup 1)\right)^*
\end{aligned}$$

Satz von Kleene

- Wir haben gezeigt, dass die von endlichen Automaten akzeptierten Sprachen genau die regulären Sprachen sind.
- Dies wird auch als der **Satz von Kleene** bezeichnet.

Satz (Satz von Kleene):
Die von endlichen Automaten akzeptierten Sprachen sind genau die regulären Sprachen.

Frage: Was können endliche Automaten nicht?

Frage: Was können endliche Automaten nicht?

Beispiel:
Die Sprache L der korrekten Klammerausdrücke über $\Sigma = \{(,)\}$.
Etwa

$$\big(()()\big), \Big(()()\big(()\big)\Big) \in L \qquad ((()), (()))()(\notin L$$

Frage: Was können endliche Automaten nicht?

Beispiel:
Die Sprache L der korrekten Klammerausdrücke über $\Sigma = \{(,)\}$.
Etwa

$$\big(()()\big), \Big(()()\big(()\big)\Big) \in L \qquad ((()), (()))()(\notin L$$

- Die Klammerung ist genau dann korrekt, wenn w gleich viele öffnende wie schließende Klammern enthält, und wenn man w von links nach rechts liest, so gibt es nie mehr „)" als „(" bis dahin.
- Ein Automat, der L erkennen kann, muss in der Lage sein, sich für ein beliebiges Wort $w \in L$ die Anzahl von (gegenüber) zu merken.
- Dies kann aber beliebig groß werden, und der Automat müsste über unendliche viele Zustände verfügen.
- Die Sprache der Klammerausdrücke ist also zwar simpel, aber wohl nicht regulär.

Pumping-Lemma für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^ix \in L$ ist für alle $i \in \mathbb{N}_0$.

Pumping-Lemma für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^ix \in L$ ist für alle $i \in \mathbb{N}_0$.

Beweis:

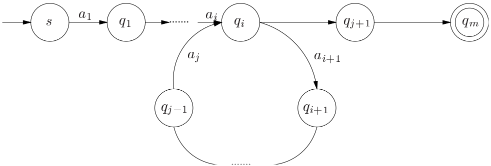
- Sei L eine reguläre Sprache.
- Dann existiert ein endlicher Automat, der L akzeptiert.
- Sei Q dessen Zustandsmenge und $n := |Q|$.
- Sei $w \in L$ mit $|w| > n$, etwa $w = a_1 \ldots a_n \ldots a_m$ mit $m > n$.

Pumping-Lemma für reguläre Sprachen

Beweis:

- Sei L eine reguläre Sprache.
- Dann existiert ein endlicher Automat, der L akzeptiert.
- Sei Q dessen Zustandsmenge und $n := |Q|$.
- Sei $w \in L$ mit $|w| > n$, etwa $w = a_1 \dots a_n \dots a_m$ mit $m > n$.

Bei der Abarbeitung von w werden dann die Zustände $q_0, \dots, q_m$ durchlaufen mit $q_m \in F$.
Dann gibt es i, j mit $0 \leq i, j \leq n$ und $i \neq j$, so dass $q_i = q_j$. Œ gelte $i < j$.

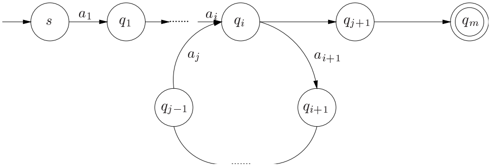


Pumping-Lemma für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^ix \in L$ ist für alle $i \in \mathbb{N}_0$.



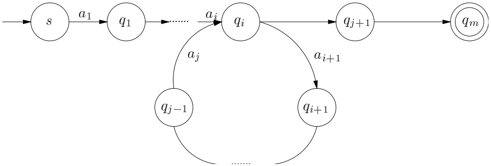
Dann kann der Zykel $q_i, q_{i+1}, \ldots, q_j = q_i$ auch gar nicht oder beliebig oft bei der Abarbeitung eines Wortes aus L durchlaufen werden so dass der Zustand $q_m \in F$ erreicht wird.

Pumping-Lemma für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^ix \in L$ ist für alle $i \in \mathbb{N}_0$.



Also gibt es eine Zerlegung $w = \underbrace{(a_1 \ldots a_i)}_{u} \cdot \underbrace{(a_{i+1} \ldots a_j)}_{v} \cdot \underbrace{(a_{j+1} \ldots a_m)}_{x}$

mit $|uv| \leq n$ und $v \neq \varepsilon$, so dass auch $uv^ix \in L$ für alle $i \in \mathbb{N}_0$.

Bemerkung

- Das Pumping-Lemma liefert nur eine notwendige, aber nicht hinreichende Bedingung für die Regularität von Sprachen.

Beispiel (1) zum PL

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^ix \in L$ ist für alle $i \in \mathbb{N}_0$.

Gegeben sei

- $\Sigma = \{0, 1\}$
- $L = \{w \in \Sigma^* \mid w \text{ enthält 10 nicht als Teilwort}\} = 0^*1^*$

Betrachte

- $n = 1, \qquad w = uvx, \qquad u = \varepsilon$

Dann

- entspricht v also dem ersten Buchstaben von w
- kann uv^ix auch 10 nicht als Teilwort besitzen.

Beispiel (2) zum PL

Karlsruher Institut für Technologie

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n\,,\ v \neq \varepsilon,$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Sei $\Sigma = \{0, 1\}$ und $L = \{0^i 1^i \mid i \geq 0\}$. Wir zeigen: L ist nicht regulär.

- Für ein n wähle $w = 0^n 1^n$
- Dann ist $|w| > n$
- Für jede Darstellung $w = uvx$ mit $|uv| \leq n$ und $v \neq \varepsilon$ ist aber

$$uv^0 x = 0^l 1^n \notin L \qquad (l < n)$$

Beispiel (3) zum PL

Sei $\Sigma = \{0, 1\}$ und

$$L = \left\{ w \in \Sigma^* \;\middle|\; w = 1^k \ (k > 0) \text{ oder } w = 0^j 1^{k^2} \ (j \geq 1,\ k \geq 0) \right\}.$$

Dann erfüllt L die Darstellung des PL:

- Sei $n = 1$ und $w \in L$ mit $|w| > 1$.
- w habe eine Darstellung $w = uvx$ mit $|uv| \leq n$ und $v \neq \varepsilon$.

Setze $u = \varepsilon$ und $|v| = 1$ das erste Symbol von w.

- Falls $w = 1^k$, so ist auch $uv^i x$ vom Typ $1^\ell \in L$.
- Falls $w = 0^j 1^{k^2}$, so ist auch $uv^0 x \in L$ (für $j = 1$ ist $uv^0 x = x = 1^{k^2}$). Für $i \geq 1$ gilt $uv^i x = 0^{j+i} 1^{k^2} \in L$.

Trotzdem ist L nicht regular. Dies lässt sich mit dem verallgemeinertem Pumping Lemma zeigen.

Verallgemeinertes PL für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| \geq n$ und jede Darstellung $w = tyx$ mit $|y| = n$ gilt:
Für das Teilwort y existiert eine Darstellung $y = uvz$ mit $v \neq \varepsilon$ bei der auch $tuv^izx \in L$ ist für alle $i \in \mathbb{N}_0$.

Verallgemeinertes PL für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| \geq n$ und jede Darstellung $w = tyx$ mit $|y| = n$ gilt:
Für das Teilwort y existiert eine Darstellung $y = uvz$ mit $v \neq \varepsilon$ bei der auch $tuv^izx \in L$ ist für alle $i \in \mathbb{N}_0$.

Beweis:

- Sei L eine reguläre Sprache.
- Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ der deterministische endliche Automat, der L erkennt.
- Setze $n := |Q| + 1$.
- Sei $tyx \in L$ mit $|y| = n$.
- Sei $q_0, \ldots, q_n$ die Folge der Zustände, die bei der Abarbeitung von y durchlaufen werden.

Verallgemeinertes PL für reguläre Sprachen

Karlsruher Institut für Technologie

Beweis:

- Sei L eine reguläre Sprache.
- Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ der deterministische endliche Automat, der L erkennt.
- Setze $n := |Q| + 1$.
- Sei $tyx \in L$ mit $|y| = n$.
- Sei $q_0, \ldots, q_n$ die Folge der Zustände, die bei der Abarbeitung von y durchlaufen werden.

- Es enthält $q_0, \ldots, q_n$ mindestens einen Zykel
- Es gibt Zerlegung $y = uvz$ so dass v der Buchstabenfolge entspricht, die beim Durchlaufen des Zykels abgearbeitet wird.
- Insbesondere ist v nicht leer.
- Dieser Zykel kann dann beliebig oft durchlaufen werden.
- Also ist auch tuv^izx ist ein gültiges Wort, das der Automat erkennt.

Verallgemeinertes PL für reguläre Sprachen

Satz:
Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| \geq n$ und jede Darstellung $w = tyx$ mit $|y| = n$ gilt:
Für das Teilwort y existiert eine Darstellung $y = uvz$ mit $v \neq \varepsilon$ bei der auch $tuv^izx \in L$ ist für alle $i \in \mathbb{N}_0$.

- Es enthält $q_0, \ldots, q_n$ mindestens einen Zykel
- Es gibt Zerlegung $y = uvz$ so dass v der Buchstabenfolge entspricht, die beim Durchlaufen des Zykels abgearbeitet wird.
- Insbesondere ist v nicht leer.
- Dieser Zykel kann dann beliebig oft durchlaufen werden.
- Also ist auch tuv^izx ist ein gültiges Wort, das der Automat erkennt.

Kapitel

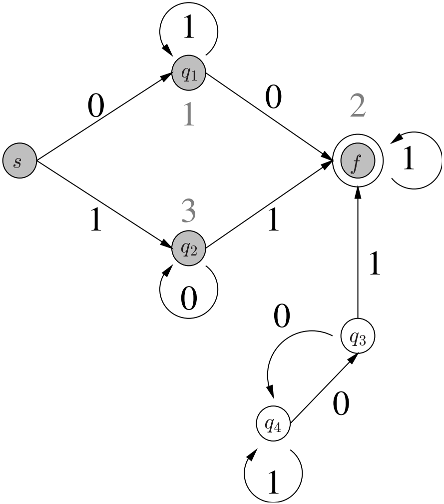
- Minimierung von Automaten
- Äquivalenzklassenautomat

Frage: Kann man konstruktiv die Anzahl der Zustände eines deterministischen endlichen Automatens erheblich verringern?

Frage: Kann man konstruktiv die Anzahl der Zustände eines deterministischen endlichen Automatens erheblich verringern?

Definition:
Zustände eines (deterministischen) endlichen Automatens, die vom Anfangszustand aus nicht erreichbar sind, heißen **überflüssig**.

Beispiel



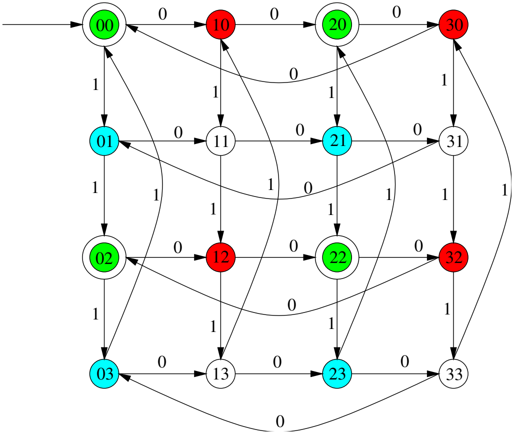
- Wir können endliche Automaten als gerichtete Graphen auffassen.
- Die überflüssigen Zustände entsprechen dann den Knoten, zu denen es vom Anfangsknoten aus keinen gerichteten Weg gibt.
- Eine Tiefensuche (**D**epth–**F**irst **S**earch, DFS) in dem Graphen liefert damit alle nicht überflüssigen Zustände.

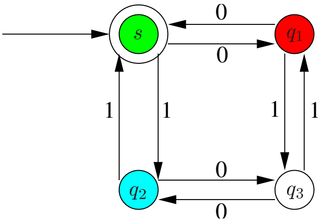
> **Satz**:
> Die Menge aller überflüssigen Zustände eines (deterministischen) endlichen Automaten kann in der Zeit $\mathcal{O}(|Q| \cdot |\Sigma|)$ berechnet werden.

Beweis: Wende DFS ab dem Startzustand an. Dies erfordert einen Aufwand proportional zu der Anzahl der Kanten in dem Graphen.

- Ein deterministischer endlicher Automat ohne überflüssige Zustände muss jedoch noch nicht minimal sein.

Beispiel





Beide Automaten akzeptieren die Sprache

$$L = \{w \in \{0,1\}^* \mid (|w|_0 \mod 2) = (|w|_1 \mod 2) = 0\}$$

 mit $|w|_a$ = Anzahl der Vorkommen des Zeichens $a \in \Sigma$ in w

Äquivalenz

Karlsruher Institut für Technologie

- Zwei Zustände haben dasselbe Akzeptanzverhalten, wenn es für das Erreichen eines Endzustandes durch Abarbeiten eines Wortes w unerheblich ist, aus welchem der beiden Zustände wir starten.
- Reduktion der Anzahl der Zustände durch Zusammlegen der Zustände mit gleichem Akzeptanzverhalten
- Letzten Beispiel: Färbung der Zustände mit gleichem Verhalten durch gleiche Farben

Definition (Äquivalenz):
Zwei Zustände p und q eines deterministischen endlichen Automaten heißen **äquivalent** ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

Offensichtlich ist $\equiv$ eine Äquivalenzrelation. Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.

Äquivalenz

Karlsruher Institut für Technologie

- Zwei Zustände haben dasselbe Akzeptanzverhalten, wenn es für das Erreichen eines Endzustandes durch Abarbeiten eines Wortes w unerheblich ist, aus welchem der beiden Zustände wir starten.
- Reduktion der Anzahl der Zustände durch Zusammlegen der Zustände mit gleichem Akzeptanzverhalten
- Letzten Beispiel: Färbung der Zustände mit gleichem Verhalten durch gleiche Farben

Definition (Äquivalenz):
Zwei Zustände p und q eines deterministischen endlichen Automaten heißen **äquivalent** $(p \equiv q)$, wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

Offensichtlich ist $\equiv$ eine Äquivalenzrelation. Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.

Der Äquivalenzklassenautomat

Definition (Äquivalenzklassenautomat):
Zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ definieren wir den Äquivalenzklassenautomaten $\mathcal{A}^{\equiv} = (Q^{\equiv}, \Sigma^{\equiv}, \delta^{\equiv}, s^{\equiv}, F^{\equiv})$ durch:

- $Q^{\equiv} := \{[q] \mid q \in Q\}$
- $\Sigma^{\equiv} := \Sigma$
- $\delta^{\equiv}([q], a) := [\delta(q, a)]$
- $s^{\equiv} := [s]$
- $F^{\equiv} := \{[f] \mid f \in F\}$

Der Äquivalenzklassenautomat

Definition (Äquivalenzklassenautomat):
Zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ definieren wir den Äquivalenzklassenautomaten $\mathcal{A}^{\equiv} = (Q^{\equiv}, \Sigma^{\equiv}, \delta^{\equiv}, s^{\equiv}, F^{\equiv})$ durch:

- $Q^{\equiv} := \{[q] \mid q \in Q\}$
- $\Sigma^{\equiv} := \Sigma$
- $\delta^{\equiv}([q], a) := [\delta(q, a)]$
- $s^{\equiv} := [s]$
- $F^{\equiv} := \{[f] \mid f \in F\}$

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ist wohldefiniert.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ist wohldefiniert.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ist wohldefiniert.

Beweis: Wir müssen zeigen, dass $F^{\equiv}$ und $\delta^{\equiv}$ wohldefiniert sind, der Rest ist klar. Dazu zeigen wir:

- ein Endzustand kann nur zu einem Endzustand äquivalent sein,
- δ führt äquivalente Zustände beim Lesen desselben Symbols wieder in äquivalente Zustände über.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ist wohldefiniert.

- ein Endzustand kann nur zu einem Endzustand äquivalent sein,

Für ε gilt:

$$\delta(p, \varepsilon) \in F \Leftrightarrow \delta(q, \varepsilon) \in F \ .$$

Es ist

$$\delta(p, \varepsilon), \delta(q, \varepsilon) \in F \text{ genau für } p, q \in F \ .$$

Also:

$$\text{Falls } p \equiv q, \text{ dann gilt } p, q \in F \text{ oder } p, q \notin F.$$

Also ist $F^{\equiv}$ wohldefiniert.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ist wohldefiniert.

- δ führt äquivalente Zustände beim Lesen desselben Symbols wieder in äquivalente Zustände über.

Sei $p \equiv q$. Dann gilt für alle $w \in \Sigma^*$

$$\delta(q, w) \in F \Leftrightarrow \delta(p, w) \in F$$

Somit gilt nach Definition von $\equiv$ auch für alle $a \in \Sigma$:

$$\delta(\delta(q, a), w) = \delta(q, aw) \in F \Leftrightarrow \delta(p, aw) = \delta(\delta(p, a), w) \in F.$$

Damit folgt $\delta(q, a) \equiv \delta(p, a)$, also ist auch $\delta^{\equiv}$ wohldefiniert.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu $\mathcal{A}$ akzeptiert dieselbe Sprache wie $\mathcal{A}$.

Der Äquivalenzklassenautomat

Satz:
Der Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu $\mathcal{A}$ akzeptiert dieselbe Sprache wie $\mathcal{A}$.

Beweis:

- Sei $w \in \Sigma^*$, $q_0 := s, q_1, \dots, q_n$ die Folge der Zustände, die von $\mathcal{A}$ bei der Abarbeitung von w durchlaufen werden.
- Bei Abarbeitung von w in $\mathcal{A}^{\equiv}$ werden dann die Zustände $[q_0], [q_1], \dots, [q_n]$ durchlaufen.
- $\mathcal{A}$ akzeptiert w genau dann, wenn $q_n \in F$ gilt. $\mathcal{A}^{\equiv}$ akzeptiert w genau dann, wenn $[q_n] \in F^{\equiv}$ gilt.
- Nach Definition von $\mathcal{A}^{\equiv}$ ist $q_n \in F$ genau dann, wenn $[q_n] \in F^{\equiv}$ gilt.

Frage

Frage:
Wie konstruiert man $\mathcal{A}^{\equiv}$ zu $\mathcal{A}$? D.h. wie berechnet man alle Äquivalenzklassen zu den Zuständen von $\mathcal{A}$?

Beweis der Äquivalenz von zwei Zuständen p scheint aufwendig:
Nach Definition muss nachgewiesen werden, dass für alle $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

- Es gibt jedoch unendlich viele $w \in \Sigma^*$.
- Es ist einfacher für p und q zu zeigen, dass p nicht äquivalent zu q ist.
- Dafür benötigen wir *nur ein* Wort $w \in \Sigma^*$ mit
  $\delta(p, w) \in F$ aber $\delta(q, w) \notin F$, oder
  $\delta(p, w) \notin F$ aber $\delta(q, w) \in F$.

Idee

Notation:
Wir bezeichnen ein solches Wort w als **Zeuge** für die Nichtäquivalenz von p und q und sagen w trennt p und q.

Idee: Teste systematisch Zustandspaare auf Nichtäquivalenz

- Betrachte alle Worte aus Σ^* in aufsteigender Länge.
- Überprüfe für jedes Wort, ob es Zeuge für Nichtäquivalenz von zwei Zuständen ist.

Frage

Frage
Wann kann dieses Verfahren abgebrochen werden?

- Sei $w = aw'$ ein *kürzester* Zeuge für $p \not\equiv q$.
- Dann ist w' Zeuge für $p' := \delta(p, a) \not\equiv \delta(q, a) =: q'$.
- Wenn es für $p' \not\equiv q'$ einen kürzeren Zeugen w'' gäbe, so wäre aw'' ein kürzerer Zeuge für $p \not\equiv q$ als w.
- Dies ist aber ein Widerspruch dazu, dass w ein kürzester Zeuge ist.

- **Fazit:** Wenn wir alle Wörter aus Σ^* in der Reihenfolge ihrer Länge darauf testen, ob sie Zeuge sind, und für eine bestimmte Länge kein Zeuge mehr für eine Nichtäquivalenz auftritt, so kann das Verfahren abgebrochen werden.

Vorgehensweise

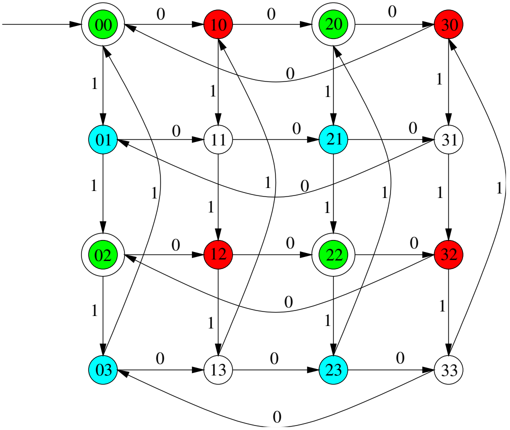
Vorgehensweise für die Konstruktion von $\mathcal{A}^{\equiv}$ aus $\mathcal{A}$

- Betrachte alle Zustandspaare und zunächst ε,
- dann alle Elemente aus Σ,
- dann alle Wörter der Länge 2 aus Σ^*,
- u.s.w.

Zunächst betrachte alle Zustände als eine Klasse.

- Dann trennt ε die Zustände aus F von denen aus $Q \backslash F$.
- Danach testen wir nur noch Paare von Zuständen aus F beziehungsweise $Q \backslash F$.
- Durch mindestens ein Wort der Länge 1 wird entweder F oder $Q \backslash F$ weiter getrennt, oder das Verfahren ist beendet.
- Dies wird iterativ so weitergeführt mit Wörtern wachsender Länge.

Beispiel zur Vorgehensweise



Vorläufige Äquivalenzklassen

vorher

$\{00, 01, 02, 03, 10, 11, 12, 13, 20, 21, 22, 23, 30, 31, 32, 33\}$

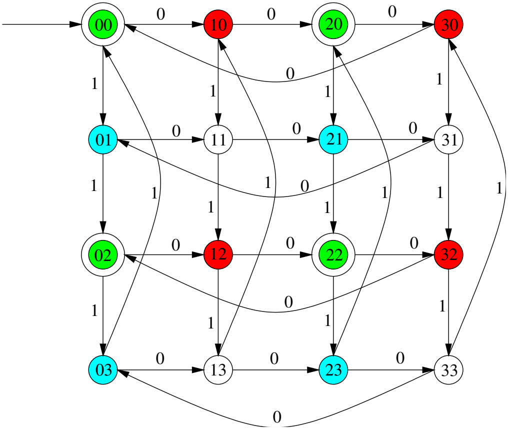
nachher

$\{00, 02, 20, 22\}$
$\{01, 03, 10, 11, 12, 13, 21, 23, 30, 31, 32, 33\}$

ε trennt $\underbrace{\{00, 02, 20, 22\}}_{\text{grün}}$ von $\{01, 03, 10, 11, 12, 13, 21, 23, 30, 31, 32, 33\}$

Beispiel zur Vorgehensweise

Karlsruher Institut für Technologie



Vorläufige Äquivalenzklassen

vorher

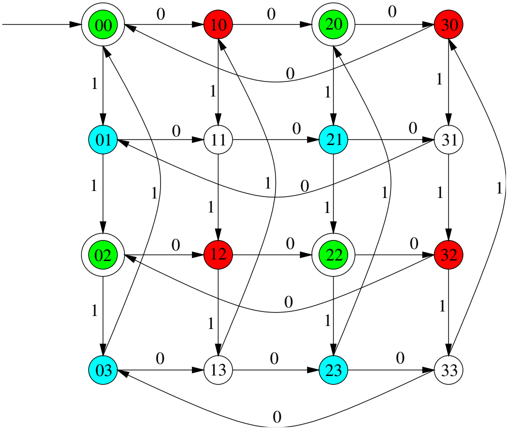
$\{00, 02, 20, 22\}$
$\{01, 03, 10, 11, 12, 13, 21, 23, 30, 31, 32, 33\}$

nachher

$\{00, 02, 20, 22\}$
$\{10, 30, 12, 32\}$
$\{01, 03, 11, 13, 21, 23, 31, 33\}$

0 trennt $\underbrace{\{10, 30, 12, 32\}}_{\text{rot}}$ von $\{01, 03, 11, 13, 21, 23, 31, 33\}$

Beispiel zur Vorgehensweise



Vorläufige Äquivalenzklassen

vorher

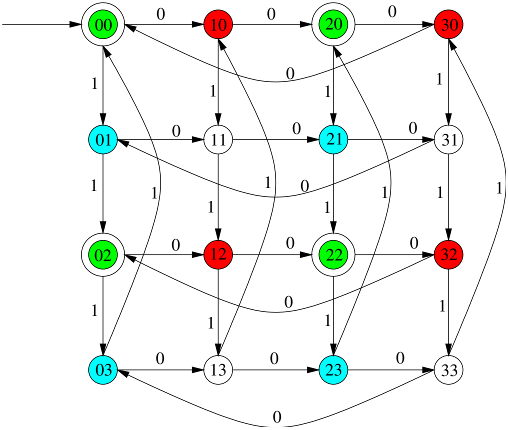
$\{00, 02, 20, 22\}$
$\{10, 30, 12, 32\}$
$\{01, 03, 11, 13, 21, 23, 31, 33\}$

nachher

$\{00, 02, 20, 22\}$
$\{10, 30, 12, 32\}$
$\{01, 03, 21, 23\}$
$\{11, 13, 31, 33\}$

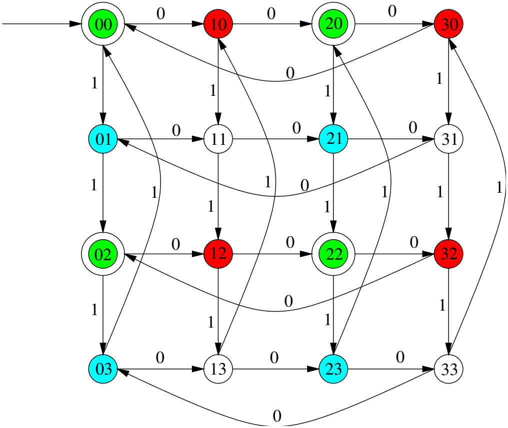
1 trennt $\underbrace{\{01, 03, 21, 23\}}_{\text{blau}}$ von $\underbrace{\{11, 13, 31, 33\}}_{\text{weiß}}$

Beispiel zur Vorgehensweise



Die Wörter 00, 01, 10, 11 trennen keine Zustandspaare mehr.

Beispiel zur Vorgehensweise



Äquivalenzklassen

$\{00, 02, 20, 22\}$
$\{10, 30, 12, 32\}$
$\{01, 03, 21, 23\}$
$\{11, 13, 31, 33\}$

Fazit: Die Äquivalenzklassen der Zustände sind:
$s = [00]$, $q_1 = [01]$, $q_2 = [10]$ und $q_3 = [11]$.

Frage

Frage:
Ist der Äquivalenzklassenautomat zu einem deterministischen endlichen Automaten schon der äquivalente Automat mit der minimalen Anzahl von Zuständen?

Frage

Frage:
Ist der Äquivalenzklassenautomat zu einem deterministischen endlichen Automaten schon der äquivalente Automat mit der minimalen Anzahl von Zuständen?

Antwort:
Ja, wir zeigen dies wie folgt:

- Zuerst konstruieren wir den minimalen Automaten zur Sprache L (Automat der Nerode-Relation)
- Anschließend zeigen wir, dass $\mathcal{A}^{\equiv}$ höchstens soviele Zustände hat wie der Automat der Nerode-Relation.

Definitionen: Rechtsinvarianz und Index

Definition (Rechtsinvarianz und Index):
Eine Äquivalenzrelation R über Σ^* heißt **rechtsinvariant**, wenn

für alle $x, y \in \Sigma^*$ gilt: falls $x\ R\ y$ so gilt auch $xz\ R\ yz$ für alle $z \in \Sigma^*$.

Den **Index** von R bezeichnen wir mit **ind(R)**; er ist die Anzahl der Äquivalenzklassen von Σ^* bezüglich R.

Nerode-Relation

Definition (Nerode-Relationen):
Für eine Sprache $L \subseteq \Sigma^*$ ist die **Nerode–Relation** R_L definiert durch: für $x, y \in \Sigma^*$ ist $x \; R_L \; y$ genau dann wenn $(xz \in L \Leftrightarrow yz \in L)$ für alle $z \in \Sigma^*$ gilt.

Die Nerode–Relation R_L zu einer Sprache $L \subseteq \Sigma^*$ ist eine rechtsinvariante Äquivalenzrelation. Es gilt:

$$
\begin{aligned}
x \; R_L \; y &\Rightarrow (xw \in L \Leftrightarrow yw \in L) \text{ für alle } w \in \Sigma^* \\
&\Rightarrow (xzw \in L \Leftrightarrow yzw \in L) \text{ für alle } w,\ z \in \Sigma^* \\
&\Rightarrow (xz \; R_L \; yz) \text{ für alle } z \in \Sigma^*.
\end{aligned}
$$

Satz von Nerode

Satz (von Nerode):
Die folgenden Aussagen sind äquivalent:

1. $L \subseteq \Sigma^*$ wird von einem deterministischen endlichen Automaten erkannt bzw. akzeptiert.
2. L ist die Vereinigung von (einigen) Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation mit endlichem Index.
3. Die Nerode–Relation hat endlichen Index.

Beweis zu Satz von Nerode: (1) $\rightarrow$ (2)

(1) $L \subseteq \Sigma^*$ wird von einem deterministischen endlichen Automaten erkannt bzw. akzeptiert.

(2) L ist die Vereinigung von (einigen) Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation mit endlichem Index.

Beweis: Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ der deterministische endliche Automat, der L akzeptiert, und $R_\mathcal{A}$ wie folgt definiert:

$$\forall\, x, y \in \Sigma^*\colon x \; R_\mathcal{A} \; y \iff \delta(s, x) = \delta(s, y).$$

- $R_\mathcal{A}$ ist eine rechtsinvariante Äquivalenzrelation.
- Der Index von $R_\mathcal{A}$ ist die Anzahl der nicht überflüssigen Zustände von $\mathcal{A}$, also endlich.
- Also ist L die Vereinigung der Äquivalenzklassen von $R_\mathcal{A}$, die zu den Endzuständen von $\mathcal{A}$ gehören.

Beweis zu Satz von Nerode: (2) $\rightarrow$ (3)

(2) L ist die Vereinigung von (einigen) Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation R mit endlichem Index.

(3) Die Nerode–Relation hat endlichen Index.

Beweis:

- Wir zeigen $x \; R \; y$ impliziert $x \; R_L \; y$ (R_L eine Vergröberung von R)
- Dann gilt $\text{ind}(R_L) \leq \text{ind}(R) < \infty$.

Sei also $x \; R \; y$.

- Da R rechtsinvariant ist, gilt für alle $z \in \Sigma^*$: $xz \; R \; yz$.
- Voraussetzung: Jede Äquivalenzklasse von R gehört entweder ganz oder gar nicht zu L
- Also: $xz, yz \in L$ oder $xz, yz \notin L$.
- Damit folgt $x \; R_L \; y$.

Beweis zu Satz von Nerode: (3) $\rightarrow$ (1)

(3) Die Nerode–Relation hat endlichen Index.

(1) $L \subseteq \Sigma^*$ wird von einem deterministischen endlichen Automaten erkannt bzw. akzeptiert.

Beweis: Wir konstruieren zu R_L einen deterministischen endlichen Automaten, der L akzeptiert. Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ mit:

- $Q := \{[x]_{R_L} \mid x \in \Sigma^*\}$, Menge aller Äquivalenzklassen bezüglich R_L.
  Es ist also $|Q| = \mathrm{ind}(R_L) < \infty$.
- $s := [\varepsilon]_{R_L}$,
- $F := \{[w]_{R_L} \mid w \in L\}$ (wohldefiniert)
- $\delta([x]_{R_L}, a) := [xa]_{R_L}$

Beweis zu Satz von Nerode: (3) $\rightarrow$ (1)

Beweis: Wir konstruieren zu R_L einen deterministischen endlichen Automaten, der L akzeptiert. Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ mit:

- $Q := \{[x]_{R_L} \mid x \in \Sigma^*\}$, Menge aller Äquivalenzklassen bezüglich R_L. Es ist also $|Q| = \text{ind}(R_L) < \infty$.
- $s := [\varepsilon]_{R_L}$,
- $F := \{[w]_{R_L} \mid w \in L\}$ (wohldefiniert)
- $\delta([x]_{R_L}, a) := [xa]_{R_L}$

δ ist wohldefiniert:

- Falls $[w]_{R_L} = [w']_{R_L}$ dann gilt $w \; R_L \; w'$ und wegen Rechtsinvarianz von R_L auch $wa \; R_L \; w'a$.
- Also ist $[wa]_{R_L} = [w'a]_{R_L}$.

Beweis zu Satz von Nerode: (3) $\to$ (1)

Beweis: Wir konstruieren zu R_L einen deterministischen endlichen Automaten, der L akzeptiert. Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ mit:

- $Q := \{[x]_{R_L} \mid x \in \Sigma^*\}$, Menge aller Äquivalenzklassen bezüglich R_L. Es ist also $|Q| = \text{ind}(R_L) < \infty$.
- $s := [\varepsilon]_{R_L}$,
- $F := \{[w]_{R_L} \mid w \in L\}$ (wohldefiniert)
- $\delta([x]_{R_L}, a) := [xa]_{R_L}$

Es bleibt zu zeigen, dass $\mathcal{A}$ genau L akzeptiert.

- Nach Konstruktion ist $\delta(s, w) = \delta([\varepsilon], w) = [\varepsilon w]_{R_L} = [w]_{R_L}$.
- Also wird w von $\mathcal{A}$ akzeptiert genau dann, wenn $[w] \in F$ gilt, d.h. wenn $w \in L$.

Korollar

Korollar
Der im dritten Beweisteil zum Satz von Nerode konstruierte Automat $\mathcal{A}$ zu R_L — der **Automat der Nerode–Relation** — ist minimal.

Korollar

Korollar
Der im dritten Beweisteil zum Satz von Nerode konstruierte Automat $\mathcal{A}$ zu R_L — der **Automat der Nerode–Relation** — ist minimal.

Beweis: Sei $\mathcal{A}' := (Q', \Sigma, \delta', s', F')$ ein deterministischer endlicher Automat, der L akzeptiert.

- Aus 1 $\Rightarrow$ 2 folgt, dass eine rechtsinvariante Äquivalenzrelation $R_{\mathcal{A}'}$ mit $\operatorname{ind}(R_{\mathcal{A}'}) \leq |Q'|$ existiert.
- Wegen 2 $\Rightarrow$ 3 gilt: $\operatorname{ind}(R_L) \leq \operatorname{ind}(R_{\mathcal{A}'})$.
- Mit 3 $\Rightarrow$ 1 folgt

$$|Q| = \operatorname{ind}(R_L) \leq \operatorname{ind}(R_{\mathcal{A}'}) \leq |Q'|,$$

für den Nerode–Automat $\mathcal{A} = (Q, \Sigma, \delta, s, F)$.

Minimalität des Äquivalenzklassenautomats

Satz (Minimalität des Äquivalenzklassenautomats):
Der Äquivalenzklassenautomat $A^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ohne überflüssige Zustände ist minimal.

Minimalität des Äquivalenzklassenautomats

Satz (Minimalität des Äquivalenzklassenautomats):
Der Äquivalenzklassenautomat $A^{\equiv}$ zu einem deterministischen endlichen Automaten $\mathcal{A}$ ohne überflüssige Zustände ist minimal.

Beweis: Sei L die vom Automaten $\mathcal{A}$ bzw. $\mathcal{A}^{\equiv}$ akzeptierte Sprache.

- $A^{\equiv}$ hat keine überflüssigen Zustände.
- Letzter Korollar: Es genügt zu zeigen, dass $|Q^{\equiv}| = \mathrm{ind}(\mathrm{R}_L)$.
- Es bleibt zu zeigen, dass für alle $x, y \in \Sigma^*$ gilt:
  $x\ R_L\ y \Rightarrow \delta(s, x) \equiv \delta(s, y)$.

$$\begin{aligned}
x\ R_L\ y &\Rightarrow \forall\, z \in \Sigma^*:\ (xz \in L \Leftrightarrow yz \in L)\\
&\Rightarrow \forall\, z \in \Sigma^*:\ (\delta(s, xz) \in F \Leftrightarrow \delta(s, yz) \in F)\\
&\Rightarrow \forall\, z \in \Sigma^*:\ (\delta(\delta(s, x), z) \in F \Leftrightarrow \delta(\delta(s, y), z) \in F)\\
&\Rightarrow \delta(s, x) \equiv \delta(s, y)
\end{aligned}$$

Zusammenfassung

- Ein DEA ist ein Modell für einen sehr einfachen Computer
- Folgende Mengen sind gleich
  - Die Menge der regulären Sprachen
  - Die Menge aller Sprachen, die von einem DEA erkannt werden.
  - Die Menge aller Sprachen, die von einem NEA erkannt werden.
- Mit Potenzmengenkonstruktion kann ein zu einem NEA äquivalenter DEA konstruiert werden.
- Das Pumping Lemma für reguläre Sprachen und das Verallgemeinerte Pumping-Lemma für reguläre Sprachen sind Hilfsmittel, mit denen für manche Sprachen gezeigt werden kann, dass sie nicht regulär sind.
- Der Äquivalenzklassenautomat zu einem DEA ohne überflüssige Zustände akzeptiert die gleiche Sprache und ist zustandsminimal.
- Der Automat der Nerode-Relation zu einem DEA akzeptiert die gleiche Sprache und ist zustandsminimal